DENSO Robotics

THIRD PARTY PRODUCTS

【サードパーティ プロダクト】

メーカ

コグネックス（株）製

製品／シリーズ

ビジョンセンサ

MODEL:In-Sight シリーズ

はじめに

本書は、「コグネックス(株)製・ビジョンセンサ・In-Sightシリーズ」をデンソーロボットコントローラRC8シリーズと接続して使用するためのプロバイダの取扱説明書です。古いIn-Sightに関しては一部使用できない機能があります。接続機器の詳細および取扱いは、「コグネックス(株)製・ビジョンセンサ・In-Sightシリーズ」の取扱説明書をご参照ください。

ご注意：(1)取扱説明書に記載された内容以外のご使用された場合は、機能・性能の保証はできませんのでご注意ください。 (2) 本書に掲載されている会社名や製品は、一般に各社の商標または登録商標です。

本書が扱う対象製品

コグネックス(株)製　　In-Sight 5000/Microシリーズ

お願い

ご使用前に「マニュアル・安全上のご注意」をお読みいただき正しく安全にプロバイダをご使用下さい。

お客さまへ

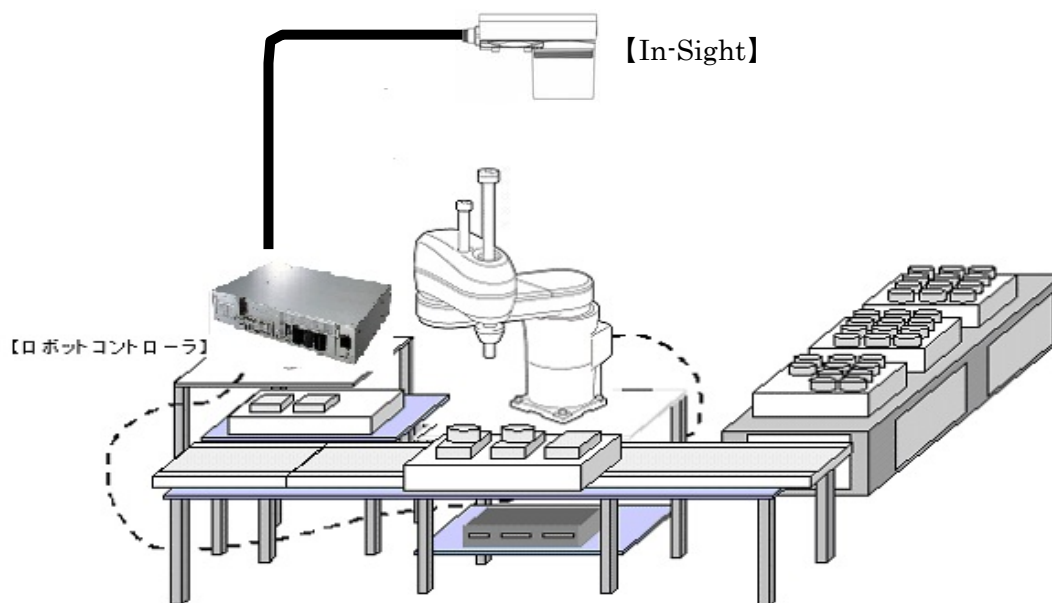
1.　ご使用に係わるリスクについて

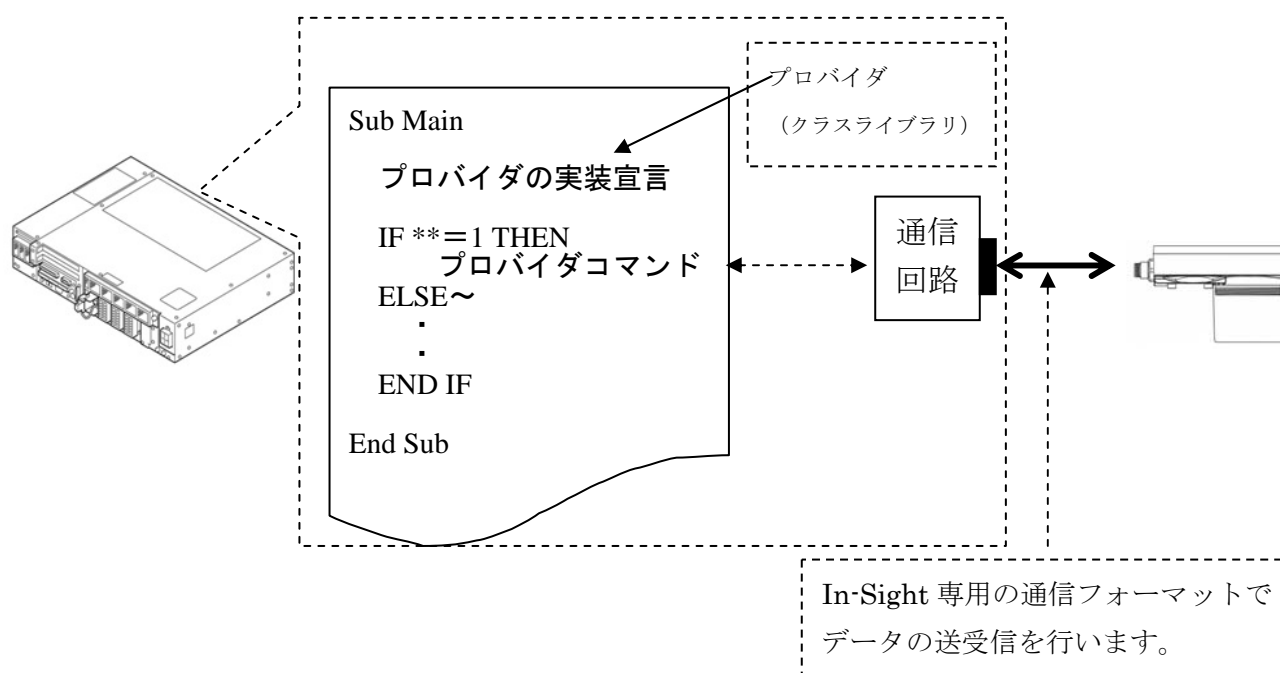
本製品（ソフトウェア）のシステム組み込み・使用ならびに本製品の使用結果に関する一切のリスクは、本製品の使用者に帰属するものとします。本製品を使用する場合は、弊社ホームページの本製品ダウンロードサイトに記載の「ソフトウェア使用許諾契約」を必ずご参照ください。

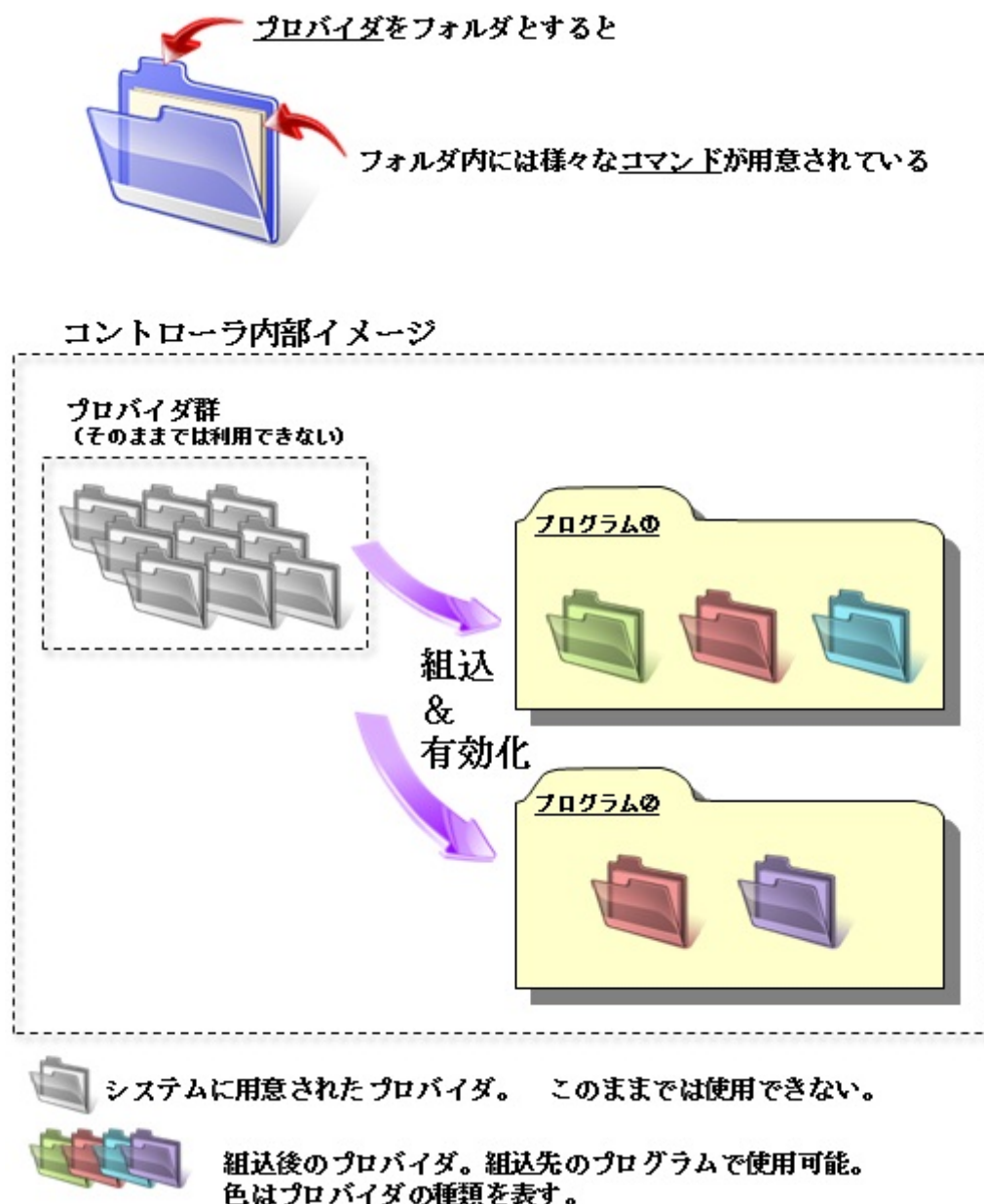
目次

1. 本製品（プロバイダ）の概要

1.1 プロバイダの対象機器

本プロバイダは、デンソーロボットコントローラ（RC8シリーズ）とIn-Sightシリーズに接続した時にのみ使用することができます。（In-Sight2000には対応していません。）

1.2 プロバイダの特長

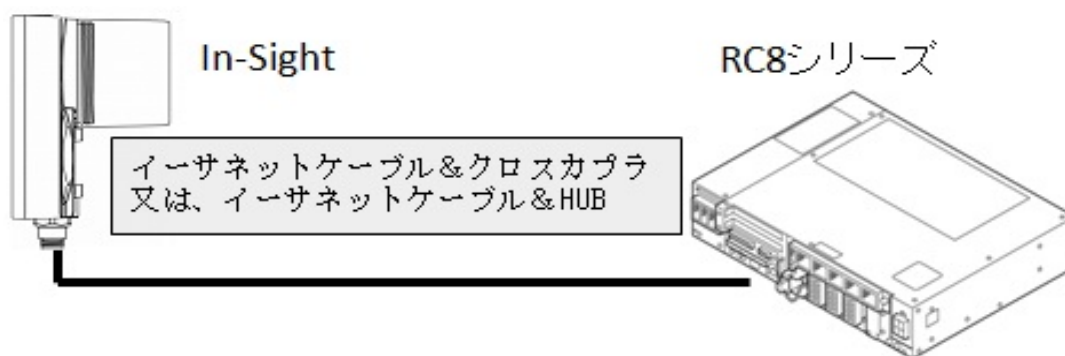
In-Sightシリーズにアクセスするために必要なIn-Sightのネイティブコマンドを、ロボットプログラムで使用できるプロバイダとして準備しています。本プロバイダを使用することで、In-Sightシリーズの通信部分のプログラムを組むことなく、容易にロボットからの通信を行うことができます。下記にプロバイダの組込関係を示します。

1.3 プロバイダの仕組み

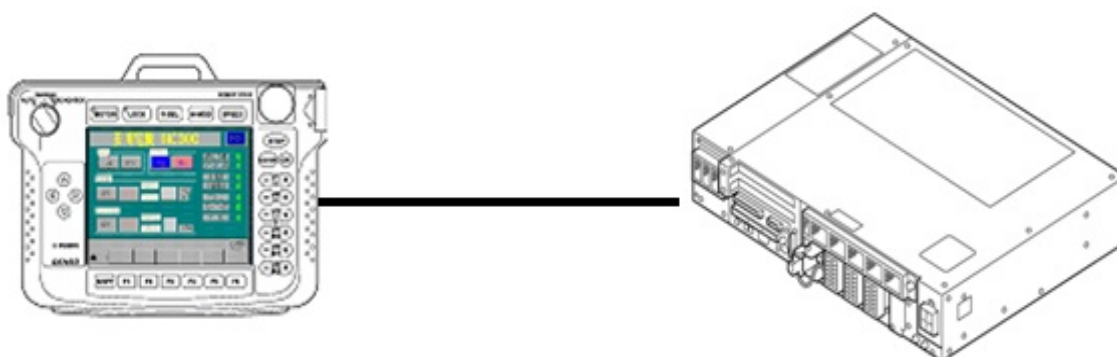
本プロバイダは対象製品の制御を行うための各種プログラムをひとつのプロバイダとして提供しています。使用にあたってはライセンスを有効化するだけで使用する事が出来ます。使用したいプログラムファイルで実装の宣言をすれば、それ以降はプロバイダが用意する関数をコマンドとしてユーザープログラム内で使用する事が可能です。本プロバイダはコントローラ内に予め用意されていますので、インストール作業は不要です。又、違う種類のプロバイダであれば複数個実装する事も可能です。但し、同じ種類のプロバイダは同じプログラム（プロシージャ）内で存在する事は出来ません。

注：上図のプロバイダ の様に、同一のプロバイダがプログラム別に存在する場合は、プログラム（タスク）間で排他処理を行う必要があります。

※プロバイダは PacScript から使用できるダイナミックリンクライブラリ（以下 DLL）として用意されています。

2. 接続方法

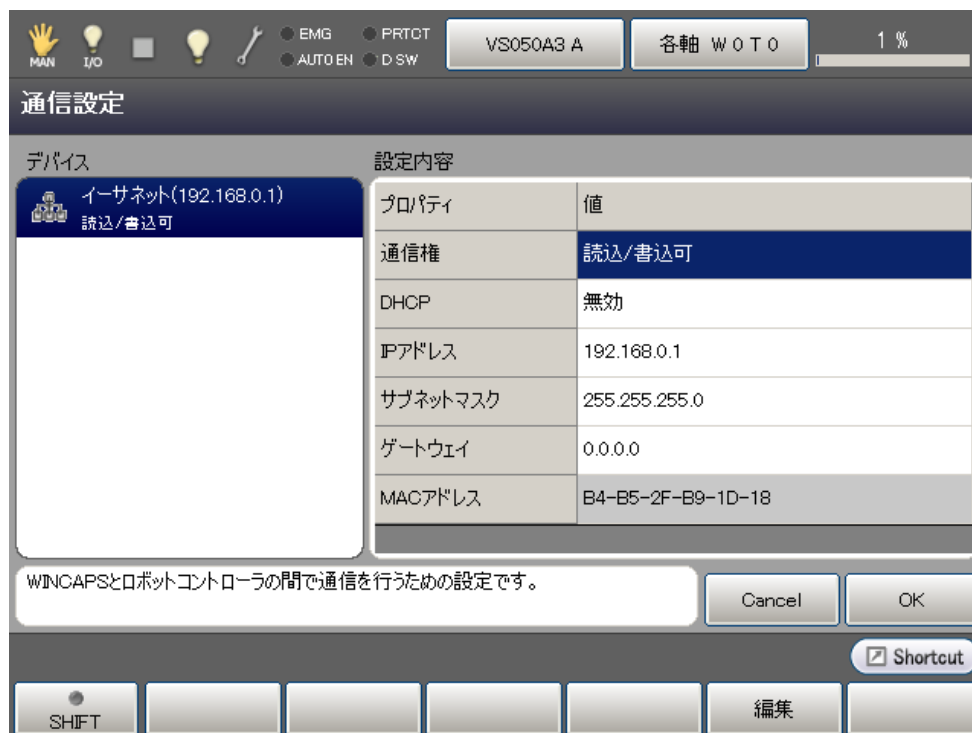
2.1 Ethernet（TCP/IP）接続例

ロボットコントローラとIn-SightをEthernet(TCP/IP)接続する際には、In-Sight付属のネットワークケーブルとクロスカプラをご使用下さい。又、スイッチングハブ/ルータを使用する場合は、スイッチングハブ/ルータの仕様に合ったケーブルをご使用下さい。

3. ロボットコントローラと使用機器の通信設定

ティーチングペンダントを使用して、各通信設定項目を使用機器に合わせて下さい。

3.1 Ethernet（TCP/IP）による通信

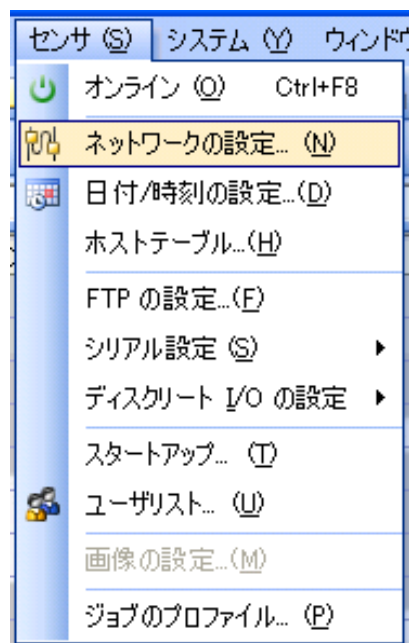
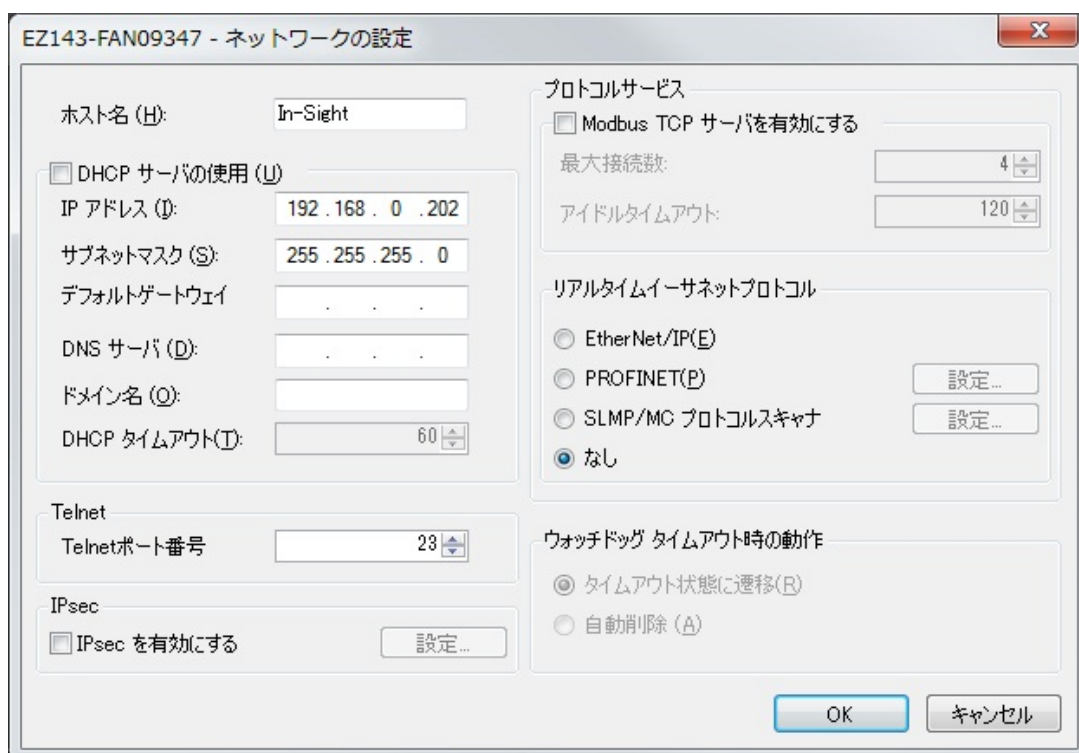
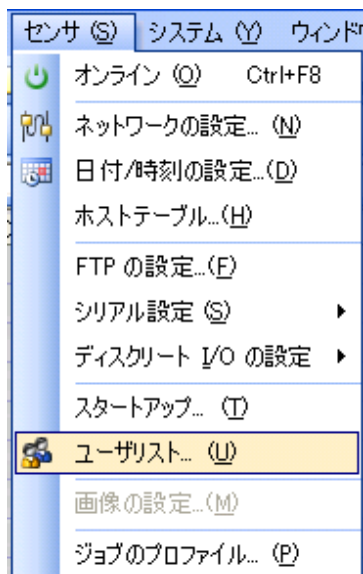
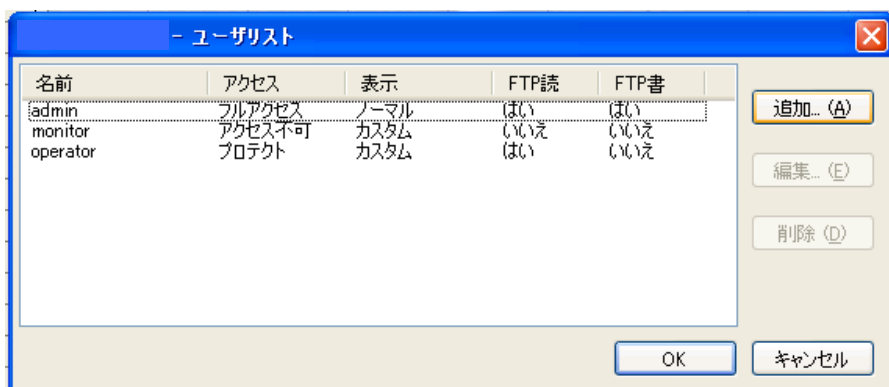
3.1.1 ロボットコントローラのEthernet（TCP/IP）通信設定

ロボットコントローラのIPアドレスを設定します。

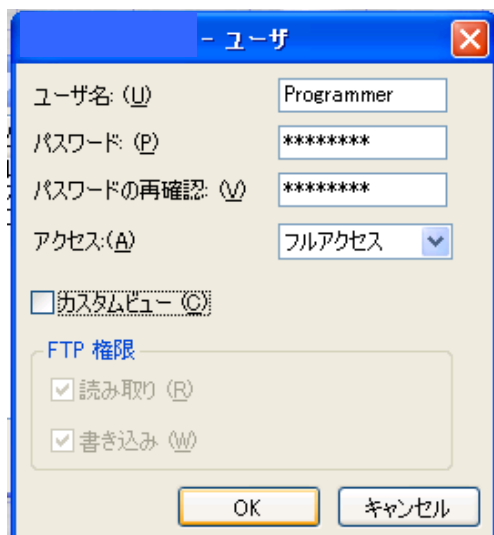
①［F6設定］［F5通信と起動権］［F2ネットワークと通信権］で、通信設定ウィンドウが表示されます。ロボットコントローラとIn-Sightが、同一のサブネットマスク内になるようにIPアドレス及び、サブネットマスクを設定して下さい。

3. 2. 1 In-SightのEthernet（TCP/IP）通信設定

In-SightのIPアドレス設定及び、ユーザリスト設定を設定します。

①[センサ]から[ネットワーク設定]選択します。

【In-Sight Explorer の画面】

②ネットワーク設定画面で IP アドレスとサブネットマスクを設定して下さい。
ロボットコントローラとIn-Sightが、同一のサブネットマスク内になるようにIPアドレス及び、サブネットマスクを設定して下さい。又、DHCPサーバは使用しないで下さい。

③[センサ]から[ユーザリスト]選択します。

④ユーザリスト画面で追加を選択します。

⑤ユーザ設定画面でユーザ名とパスワードを入力して下さい。

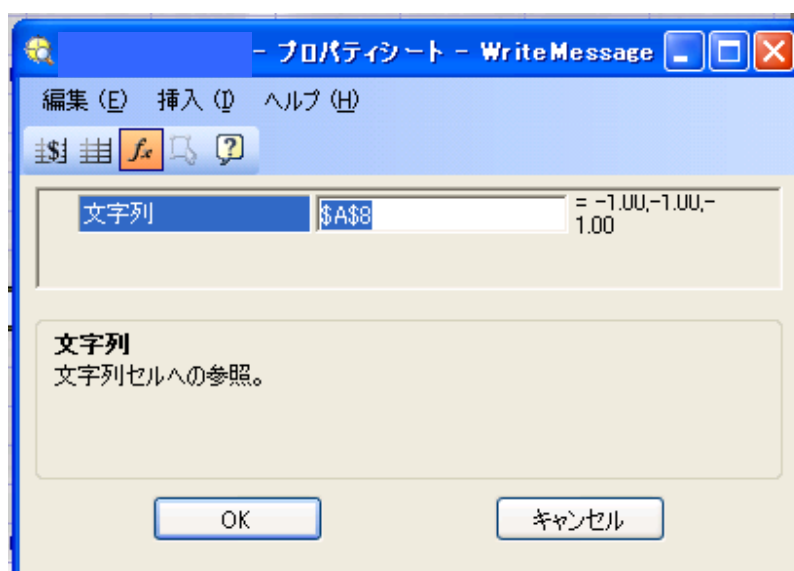
このユーザ名とパスワードは、ロボットコントローラの TCP/IP 通信用ユーザ名･パスワード登録用ファイルに登録する必要があります。

又、アクセスレベルはフルアクセスとして下さい。

3.2.2 In-SightのEthernet（TCP/IP）通信、結果出力用スプレッドシートプログラム

下記のスプレッドシートプログラム例を参考にして In-Sight Explorer にて、In-Sight のプログラミングを行って下さい。適切なプログラミングを行っていない場合、データの入出力が正常に動作されません。
なお、詳細なプログラミングについては、コグネックス社製 In-Sight ユーザーズマニュアルを参照下さい。

※データをまとめて取得する場合の Stringf、WriteMessage 関数を利用した例を記します。セルの値を個別に取得する場合は特に必要ありません。

①画像認識結果をストリング関数の Stringf にてテキスト文字列とします。

②メッセージ関数の WriteMessage にて上記のテキスト文字列（画像認識結果）を、テキスト文字列として出力します。
ネイティブ回線からデータ出力するために、WriteMessage を使用して下さい。
又、文字列参照セルは①の Stringf のセルを指定して下さい。

3.2.3 EasyBuilderでの結果取得

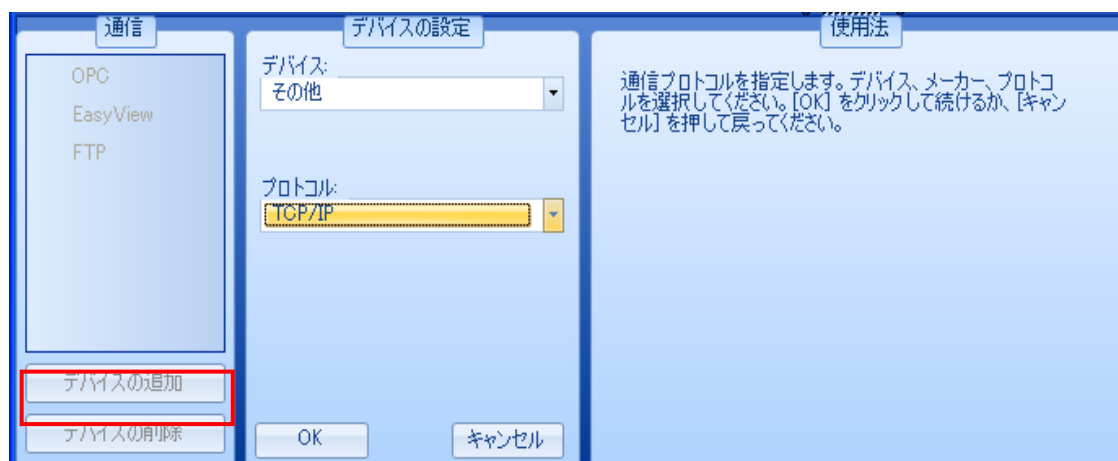
EasyBuilder で作成した画像処理結果を取得するためには、EasyBuilder の通信機能を利用します。通信設定で出力したい結果を設定し、SendMassageAndGetEZ コマンドで結果を取得したり CaoController のイベント出力として結果を受信することができます。下記に EasyBuidler で通信設定を行う方法を示します。

3.2.4 EasyBuilderの設定

始めに、結果の設定　＞　通信ボタン　を押して通信デバイスの設定画面を開きます。

次にデバイスの追加ボタン　を押して、下記のデバイスを追加してください。

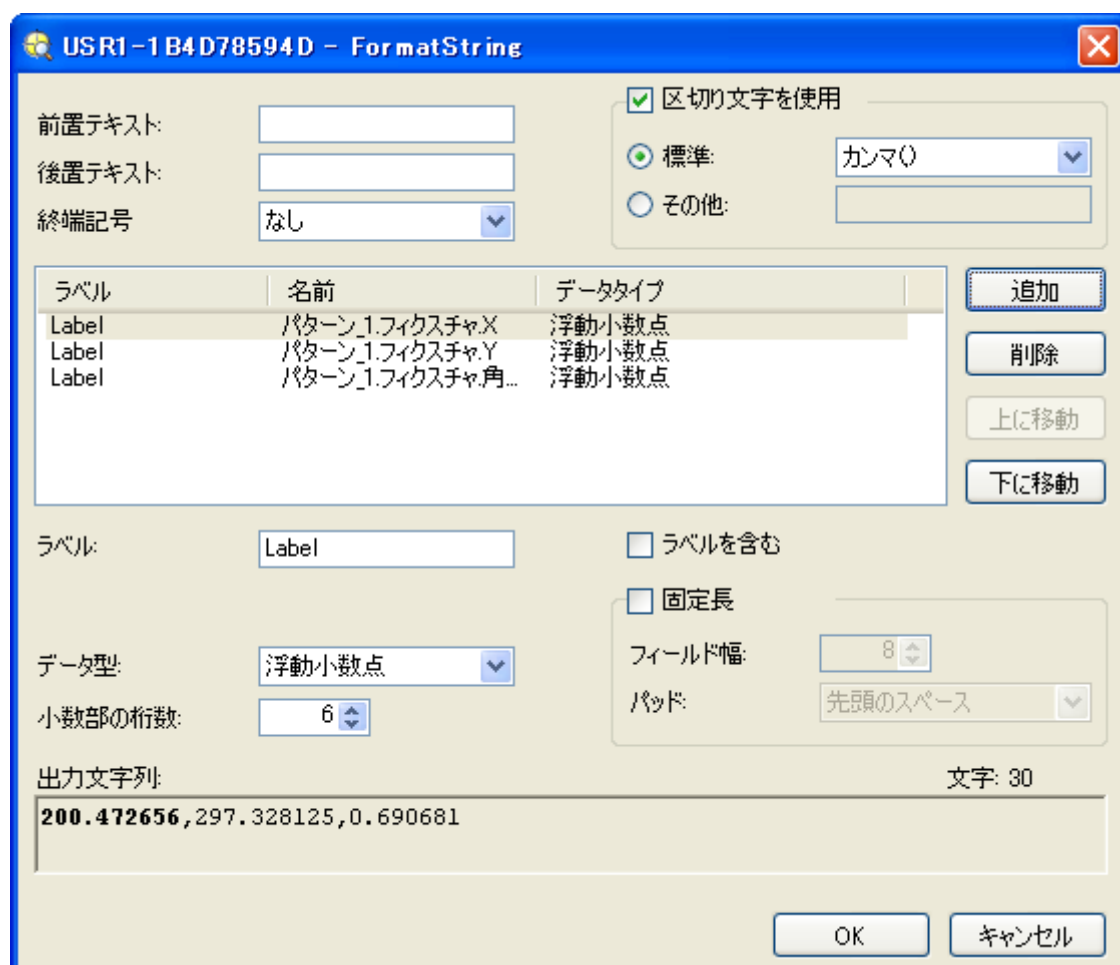
デバイス ： その他

プロトコル ：TCP/IP

次に TCP/IP の設定を行います。

サーバホスト名 ： プロバイダを使用する RC8 の IPAddress

ポート ：AddController の EZPort オプションで指定するポート番号

終端記号 ：CRLF （固定）

次にフォーマット出力文字列のタブを押し、カスタムフォーマットボタンを押して出力したい結果を登録します。その際に終端記号は「なし」にしてください。

以上の設定でオンラインモードにすると、トリガがかかった後に設定したコントローラへ結果を送信します。次に結果の受信方法を示します。

3.2.5 EasyBuilderの結果取得方法

EasyBuilder で設定した結果を受信するには、AddController のオプションに"EZPort=ポート番号"でEasybuilder に指定した出力ポートを指定する必要があります。

結果の受信コマンドは"SendMessageAndGetEZ"コマンドか"SendMessageEZ"と"GetEZ"コマンドを使用します。"SendMessageAndGetEZ"コマンドはソフトウェアトリガをかけた後、結果を受信するまで待機します。"SendMessageEZ"はソフトウェアトリガだけかけますので、その後"GetEZ"コマンドで結果を受信してください。

SendMessageAndGetEZの使用例

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202,
                    Timeout=1000, EZPort=3000")

    strResult = caoCtrl.SendMessageAndGetEZ( "Pattern2", 8, 1000 )
```

SendMessageEZとGetEZの使用例

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202,
                    Timeout=1000, EZPort=3000")

    caoCtrl.SendMessageEZ "Pattern2", 8

    '他の処理

    strResult = caoCtrl.GetEZ(1000)
```

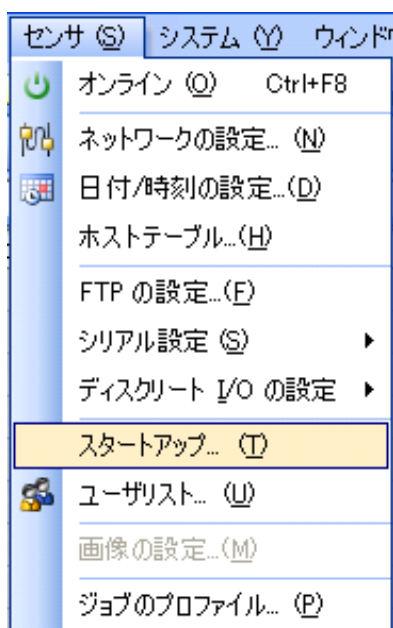
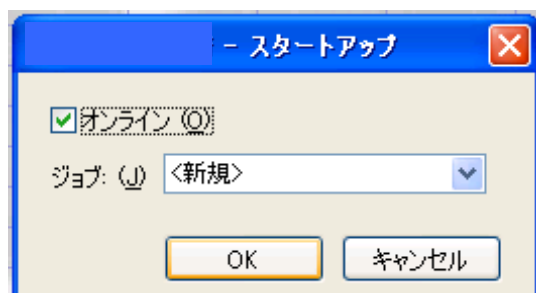
3.3 その他の設定

In-Sight は下記の条件の場合に、ネイティブモード（ロボットコントローラからの信号）でのオンライン切り替えを受け付けません。

- In-Sight Explorer 等からオフライン状態にした場合。
- ディスクリート入力でオフライン状態にした場合。
- In-Sight の電源を入れてシステムが立ち上がった時点で、オフライン状態の場合。

いずれも In-Sight Explorer やディスクリート入力でオンライン状態にすれば、ネイティブモードでのオンライン切り替えを受け付けます。
又、下記の方法で In-Sight のスタートアップをオンラインに設定することにより、In-Sight をオンラインモードで起動することができます。スタートアップの詳細については、コグネックス社製 In-Sight ユーザーズマニュアルを参照下さい。

①[センサ]から[スタートアップ]選択します。

【In-Sight Explorer の画面】

②スタートアップ設定画面で、オンラインのチェックボックスをチェックして下さい。

4. プロバイダ実行手順

プロバイダは実装（宣言）→実行が基本の手順になります。本プロバイダは実装時に接続処理を行います。操作は必要な分だけ繰り返す事が出来ます。プログラム例を下記に示します。

```
Sub Main

    On Error Goto ErrorProc ①              '異常処理ルーチンの宣言
    Dim caoCtrl as Object   ②              'プロバイダ用変数宣言
    Dim strResult as String ③              '結果取得用変数宣言

    caoCtrl =cao.AddController("InSight", "caoProv.Cognex.In-Sight", "",
                         "conn=eth:192.168.0.202") ④

    「トリガ　～　データ受信処理を記述」　⑤

EndProc:
    '終了処理
    Exit Sub

ErrorProc:
    '異常処理

End Sub
```

① 必要があればプロバイダ異常時の処理ルーチンを宣言します。（宣言時の接続異常検出）

② プロバイダを実装させる変数を Object 型で宣言します。変数名は任意に指定できます。

③ 結果を取得する変数を宣言します。データ型はコマンドにより違います。

④ プロバイダ宣言コマンド cao.AddController で実装します。設定に必要なパラメータはプロバイダで違います。これ以降は実装変数 caoCtrl を利用してプロバイダコマンドを利用できます。

⑤ これ以降は、プロバイダコマンドを使用したプログラムが記述可能です。

5. コマンドの説明

本章では各コマンドについて説明します。コマンドは接続コマンド、In-Sight コマンド、独自拡張コマンドに分類されます。尚、In-Sight コマンドの詳細動作については Cognex 社の In-Sight Explorer Reference の通信リファレンスを参照してください。

表 5-1 コマンド一覧

コマンド	In-Sight コマンド	機能	参照
接続コマンド			
cao.AddController	―	プロバイダを変数に実装して、接続処理を行います	18
Addvariable	―	画像やセルの値を取得する為の専用変数を作成します	19
Value	―	AddValiable で作成した変数を経由して取得します	20
In-Sight コマンド			
LoadFile	Load File	指定ジョブをメモリからロードし、アクティブなジョブにします	21
StoreFile	Store File	アクティブなジョブをメモリに保存します	22
DeleteFile	Delete File	メモリからジョブを削除します	23
GetFile	Get File	アクティブなジョブのファイル名を取得します	24
SetJob	Set Job	ジョブスロットからジョブをロードし、アクティブなジョブにします	25
StoreJob	Store Job	指定したジョブスロットに現在のジョブを保存します	26
DeleteJob	Delete Job	メモリ内の指定したジョブスロットからジョブを削除します	27
GetJob	Get Job	アクティブなジョブがロードされたジョブスロットを返します	28
GetValue	Get Value	指定したセルに含まれている値を返します	29
SetInteger	Set Integer	セル内に格納されている編集ボックスコントロールを、指定した整数値に設定します。	30
SetFloat	Set Float	セル内に格納されている編集ボックスコントロールを、指定した浮動小数点値に設定します。	31
SetString	Set String	セル内に格納されている編集ボックスコントロールを、指定した文字列に設定します	32
GetInfo	Get Info	In-Sight デバイスからのシステム情報を返します	33
StoreSettings	Store Settings	設定を、メモリ内の proc.set ファイルに保存します	34
SetIPLock	Set IP Lock	IP アドレスが無断で変更されることを防ぎます	35
GetIPLock	Get IP Lock	IP アドレスの アクセス不可／アクセス可能 を返します	36
SetOnline	Set Online	In-Sight デバイスをオンライン／オフラインに設定します	37
GetOnline	Get Online	In-Sight プロセッサのオンライン状態を返します	38
SetEvent	Set Event	指定したイベント (画像の取り込みなど) にトリガをかけます	39
SetEventAndWait	Set Event & Wait	指定したイベント (画像の取り込みなど) にトリガをかけ、コマンドが完了してからレスポンスを返します	40
SendMessage	Send Message	ASCII 文字列をネイティブモード接続経由で In-Sight デバイスのスプレッドシートに送信します	41
GetFilelist	Get Filelist	メモリに格納されているファイルの数と各ファイル名を返します	42
独自拡張コマンド			
NativeMode	―	ネイティブモードコマンドの送受信を行います	43
SendMessageAndWait	―	ソフトウェアトリガの発生と WriteMessage の受信を行います	44
GetMessage	―	WriteMessage の受信を行います	45

SetTimeoutNM	―	In-Sight との通信のタイムアウト時間を設定します	46
GetTimeoutNM	―	In-Sight との通信のタイムアウト時間を取得します	47
EZ Builder　用コマンド			
SendMessageAndGetEZ	―	ソフトウェアトリガの発生と EasyBuilder で設定した TCP 通信の結果出力の受信を行います	48
SendMessageEZ	―	ソフトウェアトリガの発生と GetEZ コマンドで結果を受ける準備を行います	49
GetEZ	―	EasyBuilder で設定した結果を受け取ります	50

cao.AddController

機能

プロバイダを変数に実装して、In-Sight に接続処理を行います。

書式

cao.AddController(〈コントローラ名〉,〈プロバイダ名〉,
〈プロバイダの実行マシン名〉,〈オプション〉)

引数：
〈コントローラ名〉 任意名を付けて下さい（名前で管理しています）
〈プロバイダ名〉 “CaoProv.Cognex.In-Sight”
〈プロバイダの実行マシン名〉 省略して下さい
〈オプション〉［接続パラメータ］,［ユーザ名］,［パスワード］,
［タイムアウト時間］,［ポート番号］

［接続パラメータ］ “eth=〈IP アドレス〉[:ポート番号]”
ポート番号のデフォルト：23（ポート番号は省略可）。
［ユーザー名］ In-Sight にログオンする為のユーザー名を指定します。
“Usr[=ユーザ名]”。デフォルト：admin（省略可）。
［パスワード］ In-Sight にログオンする為のパスワードを指定します。
“Password[=パスワード名]”。デフォルト：無し（省略可）。
［タイムアウト時間］ 送受信時のタイムアウト時間(msec)を指定します。
“Timeout[=時間]”。デフォルト：500（省略可）。
［ポート番号］ EasyBuilder を使用する場合の結果取得用ポート番号指定
“EZPort=ポート番号”（省略時は EasyBuilder 使用不可）。

説明

　プロバイダを変数に実装すると同時に有効にします。これ以降は実装した Object 型変数を使用してプロバイダにアクセスします。（実装された変数を“実装変数”と呼びます。）
有効と同時に telnet 接続を行います。接続の詳細は＜オプション＞で指定します。

用例

```
Dim caoCtrl as Object

caoCtrl =   cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth = 192.168.0.202”)
```

※ポート番号などを指定したい場合は次のように記述します

```
caoCtrl =   cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth = 192.168.0.202:23, Usr = admin, Password = pass, Timeout = 1000”)
```

実装変数.AddVariable

機能

画像やセルの値を取得する為の専用変数（プロパティ変数）を作成します。

書式

実装変数.AddVariable(〈変数名〉, [〈オプション〉] **)**

引数：　〈変数名〉　セル値の場合は変数名を任意文字列で、セルの場所をオプション文字列で指定します。
ネイティブモード経由で画像を取得時 @BITMAP
Datachannel 経由で画像を取得時 @BITMAP_DC
Datachannel 経由時には、オプション文字列を使用できます。

〈オプション〉　Cell = セル番号
Screen = 1(デフォルト),2,4
Port = 5000(デフォルト)
Timeout = 500(デフォルト)
SM8 = False(デフォルト)

説明

In-Sight から画像またはセルの値を取得する為の変数を作成します。
画像は配列で取得します。セル値は文字列で取得します。
Screen オプションは、設定値/Screen サイズになります。
Port は Datachannel のポート番号を指定します。
Timeout は Datachannel のタイムアウトを設定します。
SM8 は取得時に SendMessage 8 を送信しトリガをかけます。
一度 In-Sight でカメラが画像の取り込みを行わないと、AddVariable できません。
AddVariable で作成したプロパティ変数は、In-Sight の画像または固定セルよりデータを取得するための専用変数になります。同じ変数を使用して異なるセルから値を取得するには、AddVariable により変数をリダイレクトする必要があります。

用例

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetEventAndWait 8

objCell = caoCtrl.AddVariable("C40", "Cell = C40")
strResult = objCell.Value
objCell = caoCtrl.AddVariable("C41", "Cell = C41")
strResult = objCell.Value
```

プロパティ変数.Value

機能

プロパイダから取得できるデータを、AddValiable で作成した変数を経由して取得します。

書式

プロパティ変数.Value

戻り値：　AddVariable で作成した形によります。

説明

プロバイダ（実装変数）から取得できるデータを、AddVariable で作成した変数を経由して取得します。

用例

「セル値を取得する場合」

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetEventAndWait 8

objCell = caoCtrl.AddVariable(“C40”, “Cell = C40”)
strResult = objCell.Value
```

「画像を取得する場合」

```
Dim caoCtrl as Object
Dim objBmp as Object
Dim strResult as String

caoCtrl =　cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetEventAndWait 8

objBmp = caoCtrl.AddVariable(“@BITMAP”)
Label.Picture = objBmp.Value
```

実装変数.LoadFile

機能

指定したジョブをメモリからロードし、アクティブなジョブにします。

書式

実装変数.LoadFile 〈ジョブ名〉

引数：〈ジョブ名〉 ジョブ名を文字列で指定

説明

In-Sight のフラッシュメモリから、指定したジョブをロードして、アクティブジョブにします。
　In-Sight はオフラインになっている必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.LoadFile "PRO1.job"
caoCtrl.SetOnline 1
```

実装変数.StoreFile

機能

In-Sight のフラッシュメモリに現在のジョブを保存します。

書式

実装変数.StoreFile 〈ジョブ名〉

引数：〈ジョブ名〉 ジョブ名を文字列で指定

説明

In-Sight のフラッシュメモリに現在のジョブを保存します。ファイル名は引数で指定します。
In-Sight はオフラインになっている必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.StoreFile "PRO1.job"
caoCtrl.SetOnline 1
```

実装変数.DeleteFile

機能

In-Sight のフラッシュメモリから指定したジョブを削除します。

書式

実装変数.DeleteFile 〈ジョブ名〉

引数：〈ジョブ名〉 ジョブ名を文字列で指定

説明

引数で指定したジョブファイルを In-Sight のフラッシュメモリから削除します。

In-Sight はオフラインになっている必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.DeleteFile "PRO1.job"
caoCtrl.SetOnline 1
```

実装変数.GetFile

機能

In-Sight のアクティブなジョブのファイル名を返します。

書式

実装変数.GetFile

戻り値：文字列でジョブ名が返ります。

説明

ファイル名は文字列で取得します。
アクティブなジョブは保存しておく必要があります。

用例

```
Dim caoCtrl as Object
Dim strFileName as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
strFileName = caoCtrl.GetFile
```

実装変数.SetJob

機能

In-Sight のフラッシュメモリ内のジョブスロットの１つからジョブをロードし、それをアクティブにします。

書式

実装変数.SetJob ＜ジョブ ID＞

引数：〈ジョブ ID〉 ジョブ ID 番号を整数で指定します。（整数 0～999）

説明

ジョブ ID 番号を使用して、In-Sight のフラッシュメモリ内のジョブスロットの１つからジョブをロードし、アクティブなジョブにします。
ロードするジョブが数値 0～999 の接頭辞を付けて保存されている必要があります。
In-Sight はオフラインになっている必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetOnline 0
caoCtrl.SetJob 12
caoCtrl.SetOnline 1
```

実装変数.StoreJob

機能

In-Sight のフラッシュメモリ内の指定したジョブスロットに、現在のジョブを保存します。

書式

実装変数.StoreJob ＜ジョブ ID＞，　＜ジョブ名＞

引数：〈ジョブ ID〉 ジョブの ID 番号（整数 0～999）
　　　〈ジョブ名〉 ジョブの名前（文字列）

説明

In-Sight のメモリ内の指定したジョブスロットに、現在のジョブを保存します。
　In-Sight はオフラインにしておく必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetOnline 0
caoCtrl.StoreJob 3, “Test.job”
caoCtrl.SetOnline 1
```

実装変数.DeleteJob

機能

In-Sight のフラッシュメモリ内の指定したジョブスロットからジョブを削除します。

書式

実装変数.DeleteJob ＜ジョブ ID＞

引数：<ジョブ ID> ジョブの ID 番号（整数 0～999）

説明

ジョブ ID 番号で指定したジョブスロットからジョブを削除します。
In-Sight はオフラインにしておく必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.DeleteJob 3
caoCtrl.SetOnline 1
```

実装変数.GetJob

機能

In-Sight のアクティブなジョブの ID を返します。

書式

実装変数.GetJob

戻り値：整数でジョブ ID が返ります。

説明

　ジョブ ID 番号の機能を使用するには、ロードするジョブが数値 0～999 の接頭辞を付けて保存されている事が必要があります。
　このコマンドを正常に実行するために、アクティブなジョブは数値の接頭辞を付けて保存されている必要があります。

用例

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
iResult = caoCtrl.GetJob
```

実装変数.GetValue

機能

指定したセルに含まれる値を返します。

書式

実装変数.GetValue(＜列名＞，　＜行番号＞**)**

引数：〈列名〉　文字列で指定（A～Z）。
　　　〈行番号〉　整数で指定（0～399）。
戻り値：文字列でセルの値が返ります。

説明

　In-Sight のセル値に数値が含まれている場合、その数値が整数か浮動小数点かにかかわらず、小数第 3 位までにフォーマットされた浮動小数点が返されます。構造体の様な印刷されない文字がセルに含まれている場合、印刷されない文字の代わりにシャープ記号（＃）が送信されます。セルが空白の場合は、ヌル文字が送信されます。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
strResult = caoCtrl.GetValue(“B”, 14)
```

実装変数.SetInteger

機能

セル内に含まれるコントロールを、指定した整数値に設定します。コントロールは、EditInt、CheckBox、または ListBox タイプである必要があります。

書式

実装変数.SetInteger ＜列名＞， ＜行番号＞，＜設定値＞

引数：〈列名〉 文字列で指定（A～Z）。
〈行番号〉 整数で指定（0～399）。
〈設定値〉 整数で指定。

説明

〈列名〉と〈行番号〉で指定したセルの値を〈設定値〉の値に設定します。
In-Sight はオフラインにしておく必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetInteger "B", 14, 200
```

実装変数.SetFloat

機能

セル内に格納されている編集ボックスコントロールを、指定した浮動小数点に設定します。編集コントロールボックスは、EditFloat タイプである必要があります。

書式

実装変数.SetFloat　＜列名＞，　＜行番号＞，＜設定値＞

引数：〈列名〉　文字列で指定（A～Z）。
　　　〈行番号〉　整数で指定（0～399）。
　　　〈設定値〉　浮動小数点で指定。

説明

〈列名〉と〈行番号〉で指定したセルの値を〈設定値〉の値に設定します。
In-Sight はオフラインにしておく必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetInteger “B”, 14, 12.56
```

実装変数.SetString

機能

セル内に格納されている編集ボックスコントロールを、指定した文字列に設定します。編集ボックスコントロールは、EditString タイプである必要があります。

書式

実装変数.SetString　＜列名＞，　＜行番号＞，＜設定値＞

引数：〈列名〉　文字列で指定（A～Z）。
　　　〈行番号〉　整数で指定（0～399）。
　　　〈設定値〉　文字列で指定。

説明

〈列名〉と〈行番号〉で指定したセルの値を〈設定値〉の値に設定します。
In-Sight はオフラインにしておく必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetString “B”, 14, “Code”
```

実装変数.GetInfo

機能

In-Sight のセンサ情報を返します。

書式

実装変数.GetInfo

戻り値：システム情報。

説明

In-Sight のシステム情報を取得します。結果はシリアル番号、アプリケーションバージョン、モニタバージョン、MAC アドレス、ビルドの日付の順に格納されます。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
vntResult = caoCtrl.GetInfo
```

実装変数.StoreSetteings

機能

In-Sight のシステム設定を proc.set ファイルに保存します。

書式

実装変数.StoreSetteings

説明

システム設定を保存します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.StoreSetteings
```

実装変数.SetIPLock

機能

In-Sight の IP アドレスが無断で変更される事を防ぎます。

書式

実装変数.SetIPLock　＜アクセス設定＞

引数：　〈アクセス設定〉
　0　IP アドレスをアクセス可能にします。
　1　IP アドレスをアクセス不可にします。

説明

「アクセス不可」が設定されている場合、アクセス権が「プロテクト」または「アクセス不可」のユーザが「In-Sight Explorer」で行った IP アドレスの変更は保存されません。

　IP アドレスがアクセス不可の場合、アクセス権が「プロテクト」または「アクセス不可」のユーザは、In-Sight にログオン出来ますが、IP アドレスを変更する事は出来ません。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetIPLock 1
```

実装変数.GetIPLock

機能

In-Sight センサ上の IP アドレスのセキュリティステータスを返します。

書式

実装変数.GetIPLock

戻り値： 0 IP アドレスアクセス可能。
1 IP アドレスアクセス不可。

説明

IP アドレスの状態を整数で返します。。

用例

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
iResult = caoCtrl.GetIPLock
```

実装変数.SetOnline

機能

In-Sight センサをオンラインまたはオフラインに設定します。

書式

実装変数.SetOnline　＜オンライン状態＞

引数：　〈オンライン状態〉
- 0　In-Sight をオフライン設定にします。
- 1　In-Sight をオンライン設定にします。

説明

In-Sight のオンライン／オフライン状態を整数で設定します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetOnline 0
```

実装変数.GetOnline

機能

In-Sight センサのオンライン状態を返します。

書式

実装変数.GetOnline

戻り値： 0 In-Sight は現在オフラインモードです。
1 In-Sight は現在オンラインモードです。

説明

In-Sight のオンライン／オフライン状態を整数で返します。

用例

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
iResult = caoCtrl.GetOnline
```

実装変数.SetEvent

機能

In-Sight の指定したイベントにトリガをかけます。

書式

実装変数.SetEvent ＜イベントコード＞

引数： 〈イベントコード〉
0～7 ソフトウェアトリガをかけます。
8 画像取り込み、スプレッドシートを更新します。このオプションでは AcquireImage 関数トリガ引数を、カメラ、外部トリガ、または手動トリガに設定する必要があります。

説明

In-Sight にソフトウェアトリガをかけます。イベントコードは整数で指定します。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
caoCtrl.SetEvent 8
delay 1000
strResult = caoCtrl.GetValue(“C”, 13)
```

実装変数.SetEventAndWait

機能

In-Sight センサに指定したイベントをかけ、スプレッドシートの更新が完了してからレスポンスを返します。

書式

実装変数.SetEventAndWait　＜イベントコード＞

引数：　〈イベントコード〉
0～7　ソフトウェアトリガをかけます。
8　画像取り込み、スプレッドシートを更新します。このオプションでは AcquireImage 関数トリガ引数を、カメラ、外部トリガ、または手動トリガに設定する必要があります。

戻り値　なし

説明

In-Sight にソフトウェアトリガをかけます。イベントコードは整数で指定します。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetEventAndWait 8
strResult = caoCtrl.GetValue("C", 13)
```

実装変数.SendMessage

機能

ASCII 文字列をネイティブモード接続経由で In-Sight のスプレッドシートに送信します。スプレッドシートのイベントにトリガをかける事も出来ます。

書式

実装変数.SendMessage ＜設定文字列＞, ［＜イベントコード＞］

引数：〈設定文字列〉　設定する文字列を設定します。
〈イベントコード〉　SetEvent、SetEventAndWait と同様です。
省略可能です。

説明

スプレッドシート上の ReadMessage 関数に文字列を送信します。イベントコードを指定する事でイベントトリガをかける事も出来ます。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SendMessage "Pattern2", 8
delay 1000
strResult = caoCtrl.GetValue("C", 13)
```

実装変数.GetFilelist

機能

In-Sight のメモリに格納されているファイル名のリストを返します。

書式

実装変数.GetFilelist

戻り値：文字列によるファイルリスト。

説明

In-Sight のメモリに格納されているファイル名のリストを返します。In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202”)
vntResult = caoCtrl.GetFilelist
```

実装変数.NativeMode

機能

指定したネイティブモードコマンドを実行します。コマンド実行の成否に関わらずコマンドの応答を取得します。

書式

実装変数.NativeMode(＜ネイティブモードコマンド＞
，［＜オプション＞］ **)**

引数：　＜ネイティブモードコマンド＞　コマンドを文字列で指定します。
［＜オプション＞］　オプションを文字列で指定します。

戻り値：　コマンドの応答を文字列で返します。取得に失敗した場合は、文字列が格納されません。(Variant 型)

説明

指定したネイティブモードコマンドを実行します。サポートするネイティブモードコマンドは、Cognex 社の In-Sight Explorer のリファレンスマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("InSight","CaoProv. Cognex.In-Sight", "", "conn=eth:192.168.0.202")
vntResult = caoCtrl.NativeMode("GF", "BSTR=True")
```

実装変数.SendMessageAndWait

機能

SendMessage コマンドを発行した後に、In-Sight の WriteMessage 関数によって出力される文字列を受信します。

書式

実装変数.SendMessageAndWait(［<設定文字列>］
, ［<イベントコード>］
, ［<終端記号番号>］)

引数：<設定文字列>　省略した場合は空白文字が送られます。
<イベントコード>　SetEvent、SetEventAndWait と同様です。
省略した場合は８になります。
<終端記号番号>　0:無し　1:復帰(CR)　2:改行(LF)　3:CR+LF
省略した場合は１になります。

戻り値：In-Sight から出力された文字列

説明

SendMessage を実行後、WriteMessage 関数によって出力される文字列を受信待ちします。WriteMessage 関数で送られた文字列に終端記号が設定されていない場合は、タイムアウト時間まで待機し、受信できた分だけ返します。WriteMessage 関数が設定されていない場合は、タイムアウトが発生します。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
strResult = caoCtrl.SendMessageAndWait( "Pattern2", 8, 1 )
```

実装変数.GetMessage

機能

In-Sight の WriteMessage 関数によって出力される文字列を受信します。

書式

実装変数.GetMessage(［＜終端記号番号＞］ **)**

引数：＜終端記号番号＞　　0:無し　1:復帰(CR)　2:改行(LF)　3:CR+LF
省略した場合は1になります。

戻り値：In-Sightから出力された文字列

説明

In-Sight の WriteMessage 関数によって出力される文字列を受信します。
WriteMessage 関数で送られた文字列に終端記号が設定されていない場合は、タイムアウト時間まで待機し、受信できた分だけ返します。　WriteMessage 関数が設定されていない場合は、タイムアウトが発生します。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
strResult = caoCtrl.GetMessage(1)
```

実装変数.SetTimeoutNM

機能

NativeMode で通信する際のタイムアウト時間を変更します。

書式

実装変数.SetTimeoutNM ［<タイムアウト時間>］

引数：<タイムアウト時間> 設定するタイムアウト時間（msec）

戻り値：なし

説明

NativeMode で通信する際のタイムアウト時間を変更します。

初期値は AddController のオプション(Timeout)で設定した値です。

In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
caoCtrl.SetTimeoutNM 1000
```

実装変数.GetTimeoutNM

機能

NativeMode で通信する際のタイムアウト時間を取得します。

書式

実装変数.GetTimeoutNM

引数：なし

戻り値： タイムアウト時間（msec）

説明

NativeMode で通信する際のタイムアウト時間を取得します。

In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim iTimeout as Integer

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202")
iTimeout = caoCtrl.GetTimeoutNM
```

実装変数. SendMessageAndGetEZ

機能

SendMessage コマンドを発行した後に、EasyBuilder の通信（TCP/IP）によって出力される文字列を受信します。EasyBuilder の設定は 3.2.4～3.2.5 を参照して下さい。

書式

実装変数.SendMessageAndGetEZ(［<設定文字列>］
, ［<イベントコード>］
, ［<タイムアウト時間>］)

引数：<設定文字列>　省略した場合は空白文字が送られます。
<イベントコード>　SetEvent、SetEventAndWait と同様です。
省略した場合は 8 になります。
<タイムアウト時間>受信待ちする最大時間（msec）
省略した場合は 500msec 待ちます。

戻り値：In-Sight から出力された文字列

説明

SendMessage を実行後、EasyBuilder の通信設定で設定された文字列を受信します。通信設定がされていない場合や受信待ち時間を超えた場合は、タイムアウトが発生します。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202,
                  Timeout=1000, EZPort=3000”)
strResult = caoCtrl.SendMessageAndGetEZ( “Pattern2”, 8, 1000 )
```

実装変数. SendMessageEZ

機能

SendMessage コマンドを発行した後に、EasyBuilder の通信（TCP/IP）によって出力される文字列を受信する準備をします。

書式

実装変数.SendMessageEZ ［＜設定文字列＞］
，［＜イベントコード＞］

引数：＜設定文字列＞　　省略した場合は空白文字が送られます。
＜イベントコード＞　SetEvent、SetEventAndWait と同様です。
省略した場合は 8 になります。

戻り値：なし

説明

SendMessage コマンドを発行した後に、EasyBuilder の通信（TCP/IP）によって出力される文字列を受信する準備をします。文字列の受信は GetEZ コマンドを使用して下さい。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "eth=192.168.0.202,
                Timeout=1000, EZPort=3000")
caoCtrl.SendMessageEZ "Pattern2", 8
```

実装変数.GetEZ

機能

EasyBuilder の通信(TCP/IP)によって出力される文字列を受信します。

書式

実装変数.GetEZ(［<タイムアウト時間>］ **)**

引数：<タイムアウト時間>受信待ちする最大時間（msec）
省略した場合は 500msec 待ちます。

戻り値：In-Sightから出力された文字列

説明

EasyBuilder の通信(TCP/IP)によって出力される文字列を受信します。
GetEZ コマンドで結果を受信する際には、SendMessageEZ コマンドでトリガをかけて下さい。
In-Sight はオンラインになっている必要があります。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“InSight”, “CaoProv.Cognex.In-Sight”, “”, “eth=192.168.0.202,
                    Timeout=1000, EZPort=3000”)
strResult = caoCtrl.GetEZ(1000)
```

6. 操作盤画面

本プロバイダには下記の操作盤画面を準備しています。この操作盤はプロバイダを使用したもので、機器接続後の動作確認等に使用することができます。操作盤のアプリケーション例の参考にして下さい。操作盤を表示するとIn-Sightへ接続（プロバイダの実装）をしますので予め通信設定を行って下さい。操作盤を閉じると切断（プロバイダの解放）されます。

【メイン画面】

説明　各ボタンの動作内容について説明します。

1. オフライン状態へ切り替えます。(SetOnline 0)
2. オンライン状態へ切り替えます。(SetOnline 1)
3. 変更するジョブファイル名を設定します。デフォルトでは現在設定されているジョブ名を表示しています。
4. 3. で設定したジョブファイル名に切り替えます。(LoadFile)
5. In-Sight セル指定データ読み出しのセル列を設定します。範囲：A～Z
6. In-Sight セル指定データ読み出しのセル行を設定します。範囲：0～399
7. 5. 6. で設定した指定セルの列･行を In-Sight に送信し、指定セルのデータを受信して 11. のデータ表示部に表示します。(GetValue)
8. 画像取込トリガ（SetEvent 8）を In-Sight に送信します。
9. 処理結果の状態を表示します。
10. 受信データの表示ページを Up します。
11. 受信データの表示ページを Down します。

注：プロバイダの実装（初期化）が正常に行われた場合は、9. に“接続完了”と表示され、In-Sight でアクティブになっているジョブファイル名を 3. に表示します。

7. サンプルプログラム

```
Sub Main

    On Error Goto ErrProc                        '異常処理ルーチン宣言

    Dim caoCognex as Object                      'プロバイダ用変数宣言
    Dim strResult as String                      '文字列変数宣言
    Dim pTargetPos as Position                   'P 変数型変数宣言

    takearm keep = 0

    pTargetPos = P11

    caoCognex = cao.AddController("InSight", "CaoProvider.Cognex.In-Sight", "",
            "Conn=eth:192.168.0.110,Timeout = 1000")          'プロバイダの実装
    caoCognex.SetOnline 0                        'オフライン状態へ
    caoCognex.LoadFile "Check1.job"              'Check1.job ファイルに切換
    caoCognex.SetOnline 1                        'オンライン状態へ
    caoCognex.SetEventAndWait 8                  'トリガ→処理待ち
    strResult = caoCognex.GetValue("C", 31)      'スプレッドシート C31 の値を取得
    letx pTargetPos = posx(P11) + val(strResult) '受信データの X 成分を位置データへ展開
    strResult = caoCognex.GetValue("D", 31)      'スプレッドシート D31 の値を取得
    lety pTargetPos = posy(P11) + val(strResult) '受信データの Y 成分を位置データへ展開

    approach p, pTargetPos, @p 20, s = 100       '補正後の位置へ
    move l, @e pTargetPos, s = 10
    call Hand.Close
    depart l, @p 50, s = 100

EndProc:                                         '正常終了ルーチン
    「必要な終了処理を記述」
    exit sub

ErrProc:                                         '異常終了ルーチン
    「必要な異常処理を記述」

End Sub
```

※In-Sight からデータを受信するには、前述の様にセルを指定して読み出す場合とは別に、トリガ、画像処理、データ送信、をまとめて In-Sight に実行させる事も出来ます。　スプレッドシート上に ReadMessage 関数を配置しておくと SendMessageAndWait コマンドが利用できるため処理を簡素化できます。なお In-Sight への詳細な設定方法は、コグネックス社製 In-Sight ユーザーズマニュアルの SendMessage 関数を参照下さい。

```
strResult = caoCognex.SendMessageAndWait 8          'トリガ発行から受信を 1 行で実行
（ReadMessage、WriteMessage 関数がスプレッドシート上に必要です。）


caoCognex.SendMessage "Trig1", 8                    'トリガのみ
delay 1000
strResult = caoCognex.GetValue("D", 33)             'セル D33 の読み出し
（ReadMessage 関数がスプレッドシート上に必要です。）
```

改訂履歴

デンソーロボット
プロバイダ
取扱説明書
コグネックス(株)製　　ビジョンセンサ　　In-Sight シリーズ

バージョン	対応RC8	改訂内容
Ver.1.0.0	Ver.1.1.2	初版
Ver.1.0.1	Ver.1.3.6～	コマンド追加“GetMessage” 引数追加　“SendMessageAndWait”
Ver.1.0.2	Ver.1.4.*～	EasyBuilder対応、コマンド追加 SendMessageAndGetEZ, SendMessageEZ, GetEZ, SetTimeoutNM, GetTimeoutNM コマンド修正　AddController

株式会社デンソーウェーブ